YouTube: Case Study

Systems Analysis & Design

Learning Objectives

By the end of this session, you will have acquired the following information:

- YouTube System Design
- Back-of-the-Envelope Estimation
- Object Storage
- Adaptive Bitrate Sampling
- URL Shortener

YouTube 2023: Key Stats & Trends

- Total number of monthly active users: 2.7 billion.
- Number of videos watched per day: 5 billion.
- Number of creators on YouTube: 60 million.
- Amount of video content created daily: 720,000 hours.
- YouTube’s Ad revenue in 2023: $30.4 billion.

Key Requirements

- Video upload capability
- Ability to share videos.
- Smooth video streaming experience
- Option to change video quality
- Ability to like and dislike videos
- Thumbnail viewing feature
- High availability, scalability, and reliability
- Support for mobile apps, web browsers, and smart TVs

Back-of-the-Envelope Estimation

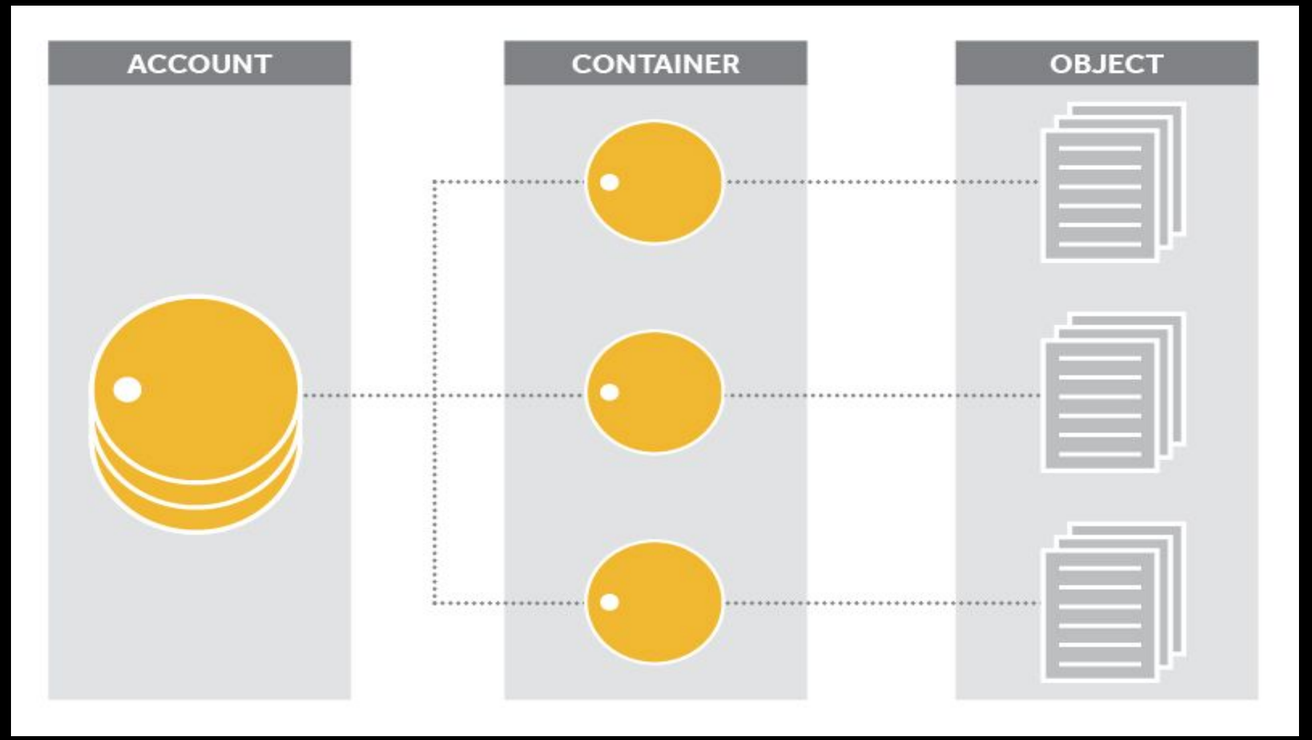
- Daily Active Users (DAU) = 2.7 billion / 30 days = 90 million
- 10% of creators upload 1 video per week = 60 million * 10% / 7 days ≈ 850,000 videos per day.
- Daily Storage Requirement: 850,000 videos at 200MB each ≈ 170TB/day * 365 days/year ≈ 62PB/year
- Write Bandwidth Requirement: 170TB/day / (24 * 60 * 60 seconds) ≈ 2 GBps
- Read Bandwidth Requirement: 5 billion/day * 200MB / (24 * 60 * 60 seconds) ≈ 12 TBps
- Number of Servers: 90 million concurrent requests / (10,000 requests/second/server) = 9,000 servers

Key Building Blocks

- Databases for storing metadata related to videos, thumbnails, comments, and user information.
- Object storage in housing all the videos on the platform.
- A Content Delivery Network (CDN) to deliver content efficiently to end users
- Load balancers for distributing the multitude of incoming client requests among the available servers.

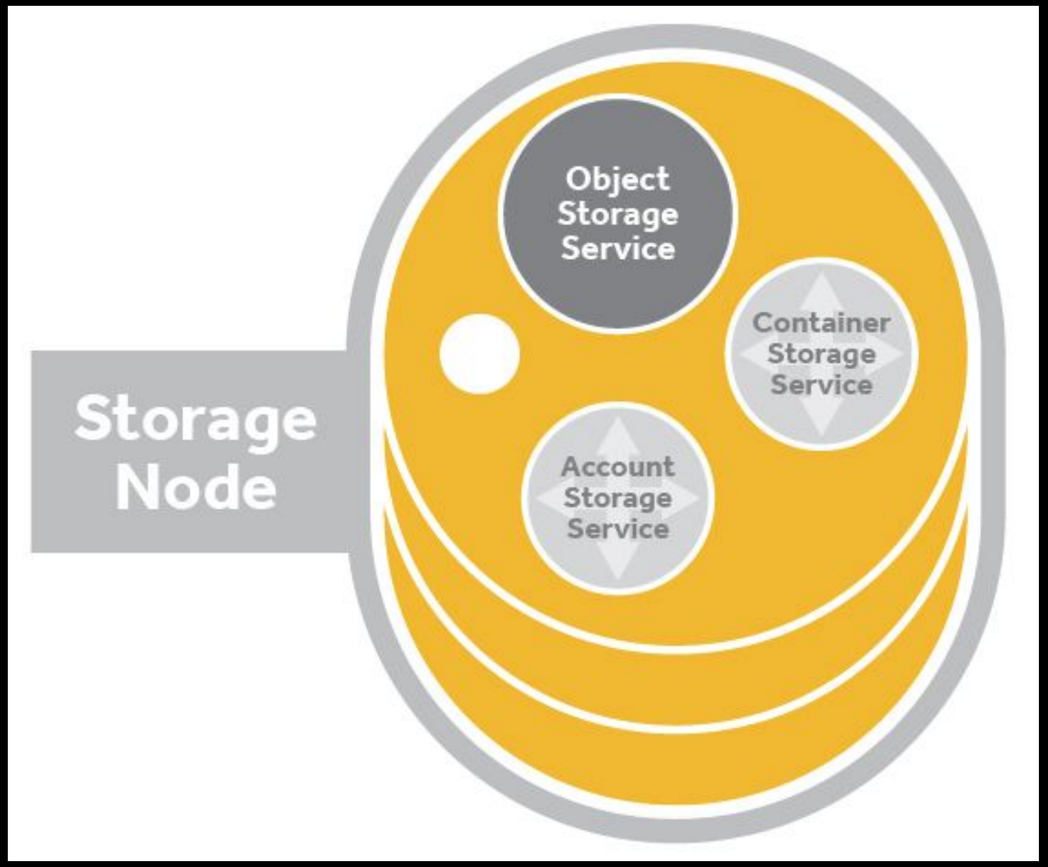
Object Storage

- Openstack Swift is a highly available, distributed, eventually consistent object/blob store.
- The Object Storage system uses a hierarchical structure:
  - **Account**: The top level of the hierarchy. Created by your service provider, it owns all resources and defines a namespace for containers. Containers can share names across different accounts.
  - **Container**: This is a namespace for objects. Each container can hold multiple objects, but an object with the same name in two different containers represents two different objects.
  - **Object**: This is used to store data content such as documents, images, and so on. Custom metadata can also be stored with an object.



- A storage URL in Swift for an object follows this format:
  - `https://swift.example.com/v1/account/container/object`
- Cluster Location: The cluster location is `swift.example.com/v1/`
- Storage Locations: `/account/container/object`
- `/account`
  - Contains metadata about the account and a list of containers in the account.
- `/account/container`
  - Stores metadata about the container and a list of objects in the container.
- `/account/container/object`
  - Where the data object and its metadata are stored.

- Swift has four server processes: proxy, account, container, and object.
- A node running only the proxy server process is known as a proxy node.
- Nodes running one or more of the other server processes (account, container, or object) are often referred to as storage nodes.

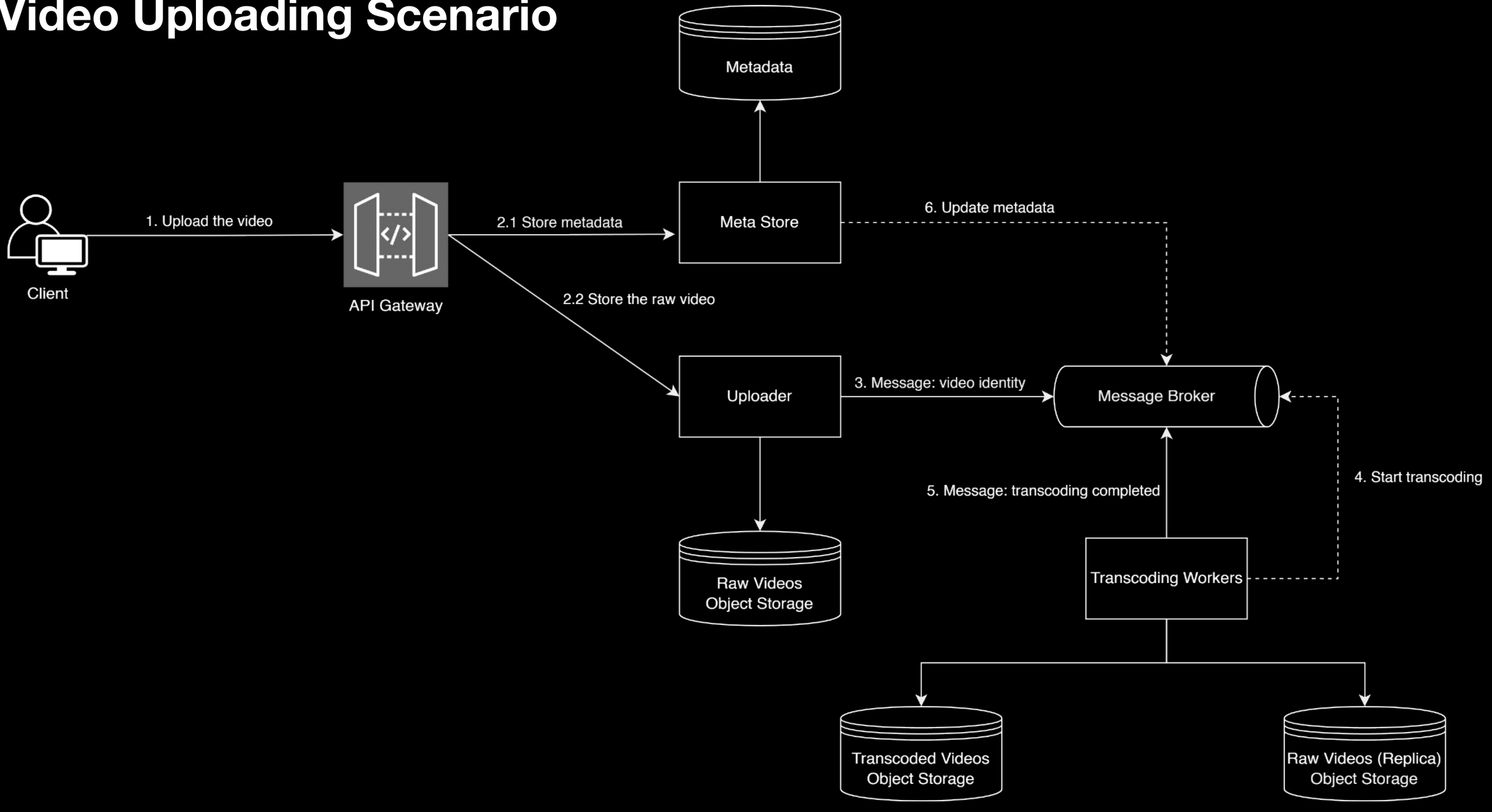


- Regions typically indicate when parts of the cluster are located within different geographical boundaries.
- Zones are defined by a distinct set of physical hardware. The failure of one zone would be isolated from other zones.
- Devices refer to the on-disk names of the devices on the server. For example, sdb1.

- Rings in the cluster determine the location of data.
- Separate rings exist for accounts, containers, and objects.
- Each consistent hashing ring is distributed to every node in the cluster.

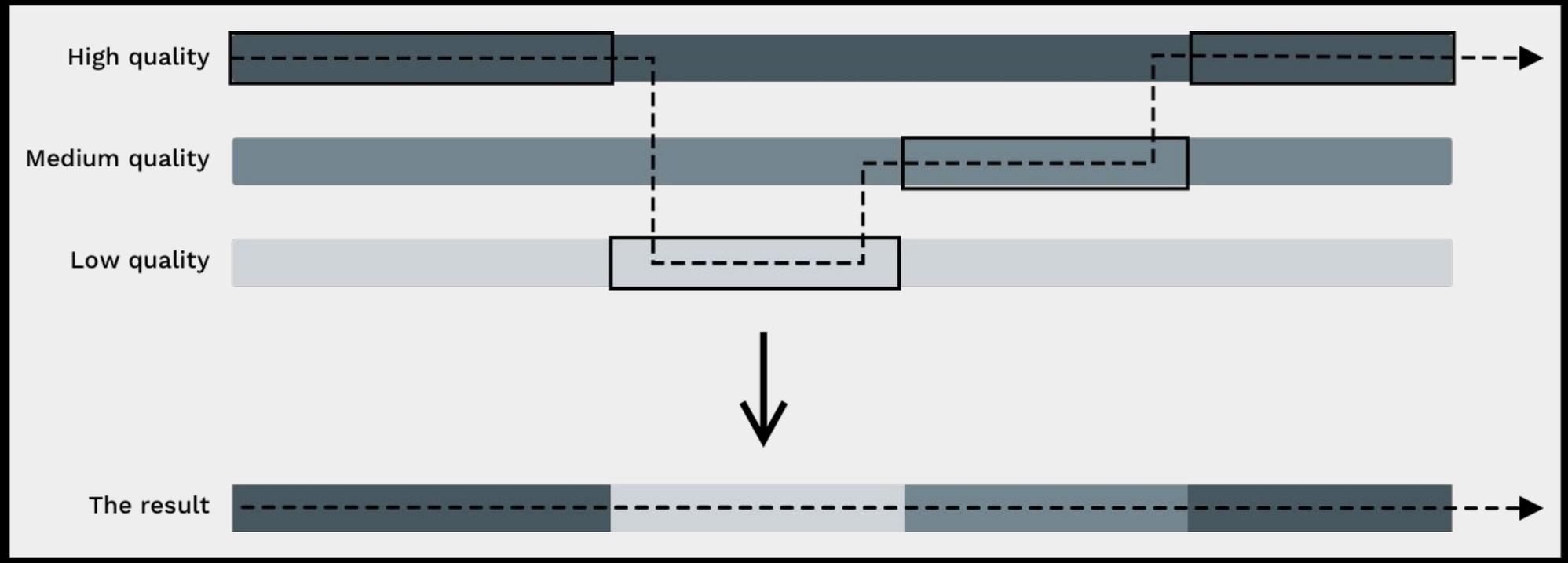
Design

Video Uploading Scenario



Adaptive Bitrate Streaming

Adaptive bitrate streaming (ABR) is used to dynamically adjust video quality based on a user’s available bandwidth, device performance, and network conditions.



Video URL

- Assuming we have the capability to generate distributed unique IDs, we could initially transform these IDs using a base 62 conversion, subsequently facilitating the creation of a URL.
- In the Twitter case study, we discuss how to create a distributed unique ID.

Unique ID	**Short ID**	**Short URL**
83929936586834022 4	b6lEW5NwxO0	https://youtu.be/b6lEW5NwxO0

Further Resources

- System Design Interview — An Insider's Guide (pages: 220 - 243)